* Writer's pictureBhuvaneswari Gopalan
  + Nov 4, 2020
  + 6 min read

**MICE algorithm to Impute missing values in a dataset**

Rated 4.667 out of 5 stars.4.7 | 3 Ratings



Missing data is a common problem in math modeling and machine learning. Machine Learning models cannot inherently work with missing data, and hence it becomes imperative to learn how to properly decide between different kinds of imputation techniques to achieve the best possible model for the use case.

Random forest (RF) missing data algorithms are an attractive approach for imputing missing data. They have the desirable properties of being able to handle mixed types of missing data, they are adaptive to interactions and nonlinearity, and they have the potential to scale to big data settings. RF imputation is found to be generally robust with performance improving with increasing correlation.

In this blog, I am going to discuss the MICE algorithm to impute missing values using Python.

**MICE** stands for ***Multivariate Imputation By Chained Equations***algorithm, a technique by which we can effortlessly impute missing values in a dataset by looking at data from other columns and trying to estimate the best prediction for each missing value. The software was published in the *Journal of Statistical Software* by Stef Van Burren and Karin Groothuis-Oudshoorn in 2011.

Rubin in 1976, classified the missing data into 3 categories:

* ***Missing Completely at Random (MCAR)*** - Implies the missingness of a field is completely random, and that we probably cannot predict that value from any other value in the data.
* ***Missing at Random (MAR)*** - Implies that the missingness of a field can be explained by the values in other columns, but not from that column.
* ***Missing NOT at Random (MNAR)*** - Implies whether there was a reason why the respondent didn’t fill up that field, and hence that data is not missing at random. For example, if someone is obese, they are less likely to disclose their weight.

If we have MNAR, we need to analyze why the data is missing, rather than straight away imputing them.

Now, let’s take an example to explain the missing values and how to fill the missing values.

Imagine a banker approaches people to take personal loans. This week he approached 6 people and jotted down the data samples he collected. Below is the collected and it has the person’s age, experience, salary and whether he took personal loan through this banker or not.



Image by Author: Dataset with missing values

Now, the banker wants to know what kind of people he should approach so that he can make more people sign up for personal loans using the data he collected.

**Step 1:** First step would be to impute the missing values in the data he collected. Let's say, the banker knows the true values for those missing values, but just wants to see how he can find out if these are missing. So, we will fill in the true values that the banker knows.

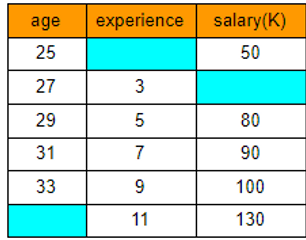
The true values of these missing values(highlighted in blue) are as shown here.



Image by Author: Dataset with true values to verify the model output

We will keep this data aside for future reference to cross verify our model if it is good or not.

**Step 2**: Remove the "Personal Loan" column as it is the target column, we will not need this column for imputation. We will just impute the 3 feature columns age, experience and salary from this matrix as shown below:



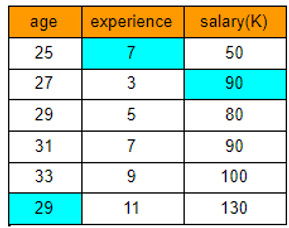
A question now arises on seeing the above feature matrix, Why not use Univariate methods like mean, median, mode, frequently used, constant etc. to impute the missing values?

*Note:*

*Univariate method uses that particular column to impute the missing values in that column. For example, mean, mode, median etc. types of imputation.*

*Multivariate method imputes missing values in a dataset by looking at data from other columns and estimating**the best prediction for each missing value.*

To answer the question, let’s apply the mean imputation method on the above feature matrix to fill the missing values. When we apply the mean imputation, we get the following output.



Just by observing the results, we see that a 25 year old has 7 years of IT work experience which is not possible and that too a 7 years experience person earning 50K is totally misleading. And similarly a 29 year old cannot have 11 years of experience.

So, we can see that mean imputation is not working as expected.

So, this "brute-force" approach is one of the issues with single imputation or univariate imputation techniques.

Multivariate technique solves this issue by factoring in other variables in the data to make better predictions of the missing values. Which means, to find the missing value in age column, we try to run a regression model on the 3 features with experience and salary as features and age as target. Similarly we will try to get the missing values from experience and salary features as well.

Okay, now let us look at how it is actually done using MICE algorithm.

We all know that MICE algorithm is an iterative imputation method. So let's see in detail each iteration and how the missing values are computed and whether the prediction is close to the true values.

**Iteration 1:**

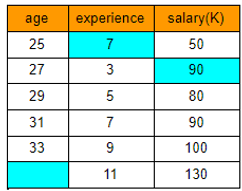
"Zeroth" dataset

**Step 1:** Impute all missing values using mean imputation with the mean of their respective columns.

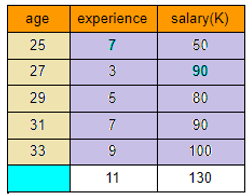
We will call this as our "*Zeroth"* dataset

Note:

We will be imputing the columns from left to right.



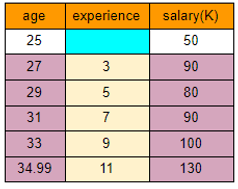
**Step 2:** Remove the "age" imputed values and keep the imputed values in other columns as shown here.



**Step 3:** The remaining features and rows(top 5 rows of experience and salary) become the feature matrix(purple cells), "age" becomes the target variable(yellow cells). We will run the linear regression model on the fully filled rows with X= experience and salary and Y=age. To estimate the missing age, we will use the missing value row (white cells) as the test data.

So, top 5 rows will be training data and the last row that has missing age will be test data. We will use (experience = 11 and salary = 130) to predict corresponding "age" value.

When I did this, I found that my model predicted the age as **34.99**.



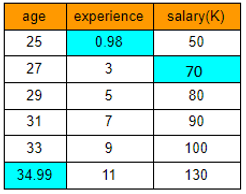
**Step 4:** Update the predicted age value in the missing cell in "age" column. Now, remove "experience" imputed value. The remaining features and rows becomes the feature matrix(purple cells) and "experience" becomes the target variable(yellow cells). We will run the linear regression model on the fully filled rows with X= age and salary and Y=experience. To estimate the missing experience, we will use the missing value row (white cells) as the test data.

The predicted value for experience is **0.98**.



**Step 5:** Update the predicted experience value in the missing cell in "experience" column. Now, remove "salary" imputed value. The remaining features and rows becomes the feature matrix(purple cells) and "salary" becomes the target variable(yellow cells). We will run the linear regression model on the fully filled rows with X= age and experience and Y=salary. To estimate the missing salary, we will use the missing value row (white cells) as the test data.

The predicted value for Salary is **70**.



We have now imputed the missing values in the original dataset and the predicted values after 1st iteration is shown here.

Let's name this as "*First*" dataset.

This is Iteration 1 done and dusted.

**Step 6:** We will subtract the two datasets(zeroth and first). The resultant dataset is as below:



If we observe, the absolute difference between 2 datasets are higher in few imputed values. Our aim is to reduce these differences close to 0. To achieve this we have to do many iterations.

So, now we repeat the steps 2-6 with the new dataset (first), until we get a stable model. i.e. until the difference between the 2 latest imputed datasets becomes very small, close to 0.

Technically, we stop the iterations when a pre-defined threshold is breached or we can do until a pre-defined maximum number of iterations gets completed.

**Iteration 2:**

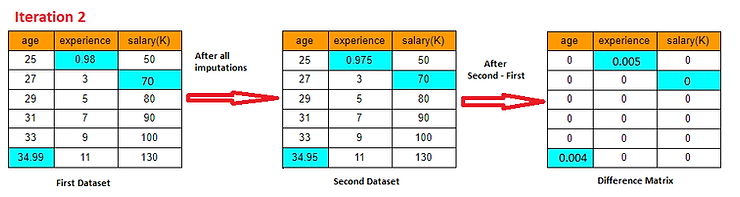


"First" dataset

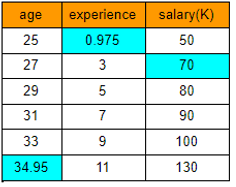
Now we will use the "first" dataset as our base dataset to do imputations, and discard the "Zeroth" dataset which had the mean imputations.

With "first" dataset as base, let's perform all the steps 2-6 and again predict the imputed values for the initial 3 missing values.

Here's is my iteration 2 values. I took the first dataset , did all the imputations and subtracted the new dataset values form first dataset and got the difference matrix as shown below:



Now, after second iteration, we can see that the difference is very negligible.



"Second" dataset

The second dataset imputed values for age is **34.95**, experience is **0.95** and Salary is **70**. When we compare these imputed values with the true values of the missing values which is age =35, experience = 1 and Salary = 70K, it is almost same with very small difference.

We can either stop here as we almost got the same numbers, or proceed with next iteration until we get 0 difference.

As far as this example is concerned, we will stop it here. And so, the second dataset will be the final imputed values for the missing values as shown in the above table.

This is in a nutshell the ***Multivariate Imputation By Chained Equations(MICE).*** I just showed you using the numerical columns, but we can also do it for categorical columns. In the case of categorical values, it will be little more complex and it will be encoded first.

**Key takeaways:**

Missing data often plagues real-world datasets, and hence there is tremendous value in imputing, or filling in, the missing values. Often the common methods like mean, median ,mode, frequent data and constant doesn't provide the correct data for the missing values.

The model is only as good as the data, so having a complete dataset with proper data is a must. Consider using MICE algorithm next time when you need to impute missing data!

The implementation of MICE algorithm using scikit-learn Iterative Imputer is explained in my next blog "[*How to implement MICE algorithm using Iterative Imputer to handle missing values?*](https://www.numpyninja.com/post/how-to-implement-mice-algorithm-using-iterative-imputer-to-handle-missing-values)".

***Happy Modeling!***